|  |
| --- |
| [公司名称] |
| 标题 |
| 副标题 |

|  |
| --- |
| 姚斯元  2018-10-30 |

目录

[一、 123 1](#_Toc3539499)

[1.1 1 1](#_Toc3539500)

[1.1.1 3 1](#_Toc3539501)

# Oracle

## 第一天:

### 1.Oracle概念和安装

Oracle的OCA认证

Oracle : 数据库 , 4 2000美元 1200美元60% 程序员 写数据 ,读出数据

处理大量数据

IBM公司发表一篇论文, R关系型数据库模型

埃里森:

### 为什么要学习Oracle:

mysql : 开源免费的数据库, 中小型企业, 淘宝网 mysql集群

Oracle: 收费的数据库, 国企, 金融, 银行, 证券,保险 , 京东 按CPU核数, 一核=永久使用费用 18万.

Oracle虚拟机安装:

1. 先安装 vm wareworkstation

2. 双击纯净版中的 xp\_oracle.vmx文件

3. 关闭防火墙

配置网络:

1. 编辑 ----> 添加虚拟网络

2. 设置虚拟机网络为固定IP地址

3. 测试网络是否ping通

Oracle软件安装:

1.将Oracle拖到虚拟机中

2.安装数据库软件

3.解锁Scott和HR账户

4.配置Oracle的地址为固定IP地址

5.重启Listener 和 ORCL服务



6.客户端测试访问ORACLE

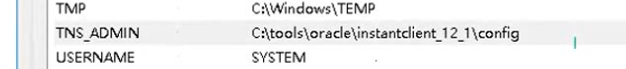


PLSQLDevelop的安装: (注意:安装在没有中文,没有空格的路径下)

1.先安装instantclient

2.配置tools --> preferences --> 配置instantclient的路径

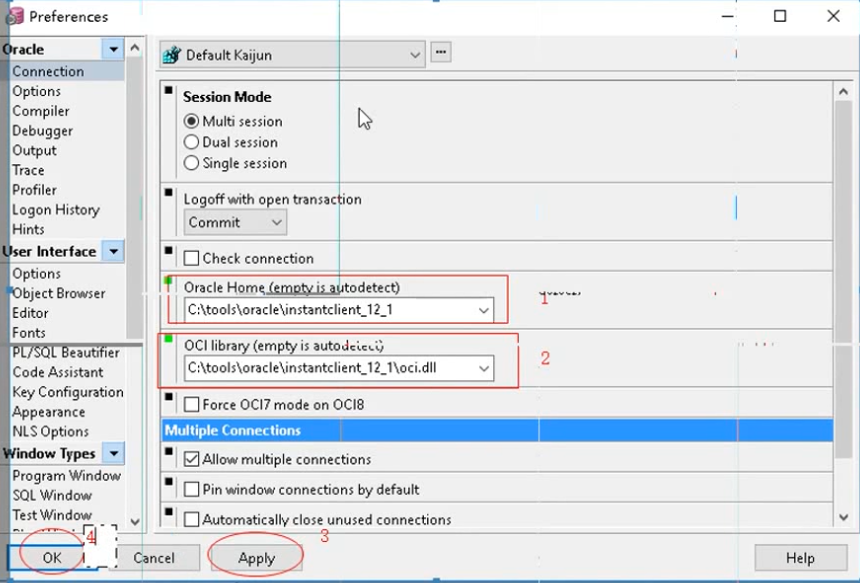
3.将前面步骤中tnsnames.ora 文件拷贝到本机, 配置TNS\_ADMIN环境变量



4.重启PLSQLDevelop

5.输入用户名和密码测试

Pl的配置



3.条件查询

4.Oracle中的函数

select \* from tab;

/\*

数据库 ---> 数据库实例 ---> 表空间(逻辑单位)(用户) ---> 数据文件(物理单位)

地球 ---> 一个国家 ---> 省份(逻辑单位)(公民) ---> 山川河流(物理单位)

通常情况下,ORacle数据库只会有一个实例ORCL,

新建一个项目:

MYSQL : 创建一个数据库,创建相应的表

Oracle: 创建一个表空间,创建用户,用户去创建表

### Oracle和MYSQL的差别

Oracle是多用户的, MYSQL是多数据库的

1. 遵循SQL标准

2. 不同厂商,不同的数据库产品,但是有自己的方言

3. 使用自己的方言,也能够完成相同的功能

4. Oracle安全级别要高,MYSQL开源免费

\*/

/\*

### 基本查询:

SQL : 结构化查询语言

请听题: 请说一下SQL的分类以及每类常见的操作符都有哪些

四类:

DDL : 数据定义语言 create alter drop truncate

DML : 数据操纵语言 insert update delete

DCL : 数据控制语言 安全 授权 grant revoke

DQL : 数据查询语言 select from子句 where子句

查询语句的结构:

select [列名] [\*] from 表名 [where 条件] [group by 分组条件] [having 过滤] [order by 排序]

\*/

select \* from emp;

select 1+1; --在Oracle等于报错 ,在MYSQL中输出结果是2

/\*

dual : oracle中的虚表 ,伪表, 主要是用来补齐语法结构

\*/

select 1+1 from dual;

select \* from dual;

select 1 from emp;

--直接写一个常量比写 \* 要高效

select count(1) from emp;

select count(\*) from emp;

/\*

别名查询: 使用as 关键字, 可以省略

别名中不能有特殊字符或者关键字, 如果有就加双引号

\*/

select ename 姓名, sal 工资 from emp;

select ename "姓 名", sal 工资 from emp;

/\*

去除重复数据 distinct

多列去除重复: 每一列都一样才能够算作是重复

\*/

--单列去除重复

select distinct job from emp;

--多列去除重复的

select distinct job,deptno from emp;

--查询中四则运算

select 1+1 from dual;

--查询员工年薪 = 月薪\* 12

select sal\*12 from emp;

--查询员工年薪+奖金

select sal\*12 + comm from emp;

--nvl 函数 : 如果参数1为null 就返回参数2

select sal\*12 + nvl(comm,0) from emp;

/\*

注意: null值 , 代表不确定的 不可预知的内容 , 不可以做四则运算

\*/

/\*

字符串拼接:

java : + 号拼接

Oracle 特有的连接符: || 拼接

在Oracle 中 ,双引号主要是别名的时候使用, 单引号是使用的值, 是字符

concat(str1,str2) 函数, 在mysql和Oracle中都有

\*/

--查询员工姓名 : 姓名:SCOTT

select ename from emp;

--使用拼接符

select '姓名:' || ename from emp;

--使用函数拼接

select concat('姓名:',ename) from emp;

### 条件查询

/\*

条件查询 : [where后面的写法]

关系运算符: > >= = < <= != <>

逻辑运算符: and or not

其它运算符:

like 模糊查询

in(set) 在某个集合内

between..and.. 在某个区间内

is null 判断为空

is not null 判断不为空

\*/

--查询每月能得到奖金的员工信息

select \* from emp where comm is not null;

--查询工资在1500--3000之间的员工信息

select \* from emp where sal between 1500 and 3000;

select \* from emp where sal >= 1500 and sal <= 3000;

--查询名字在某个范围的员工信息 ('JONES','SCOTT','FORD') in

select \* from emp where ename in ('JONES','SCOTT','FORD');

\_ 匹配单个字符

如果有特殊字符, 需要使用escape转义

\*/

/\*

模糊查询: like

% 匹配多个字符

--查询员工姓名第三个字符是O的员工信息

select \* from emp where ename like '\_\_O%';

--查询员工姓名中,包含%的员工信息

select \* from emp where ename like '%\%%' escape '\';

select \* from emp where ename like '%#%%' escape '#';

/\*

排序 : order by

升序: asc ascend

降序: desc descend

排序注意null问题 : nulls first | last

同时排列多列, 用逗号隔开

\*/

--查询员工信息,按照奖金由高到低排序

select \* from emp order by comm desc nulls last;

--查询部门编号和按照工资 按照部门升序排序, 工资降序排序

select deptno, sal from emp order by deptno asc, sal desc;

### 函数

/\*

函数: 必须要有返回值

单行函数: 对某一行中的某个值进行处理

数值函数

字符函数

日期函数

转换函数

通用函数

多行函数: 对某一列的所有行进行处理

max() min count sum avg

1.直接忽略空值

\*/

--统计员工工资总和

select sum(sal) from emp;

--统计员工奖金总和 2200

select sum(comm) from emp;

--统计员工人数 14

select count(1) from emp;

--统计员工的平均奖金 550 错误 2200/14 =

select avg(comm) from emp;

--统计员工的平均奖金 157.

select sum(comm)/count(1) from emp;

select ceil(sum(comm)/count(1)) from emp;

update emp set ename = 'TUR%NER' where ename = 'TURNER';

select \* from emp;

--数值函数

select ceil(45.926) from dual; --46

select floor(45.926) from dual; --45

floor ： 小数部分下取整，即向下取最近的整数

select floor(1.9) from dual; // 结果为1

select floor(1.0) from dual; // 结果为1

select floor(1.01) from dual; // 结果为1

select floor (-12.5) from dual; -- -13

select floor (12.5) from dual; -- 12

ceil : 小数不为零部分上取整，即向上取最近的整数

select ceil(1.9) from dual; // 结果为2

select ceil(1.0) from dual; // 结果为1

select ceil(1.01) from dual; // 结果为2

select ceil (-12.5) from dual; -- -12

select ceil (12.5) from dual; -- 12

--四舍五入

select round(45.926,2) from dual; --45.93

select round(45.926,1) from dual; -- 45.9

select round(45.926,0) from dual; --46

select round(45.926,-1) from dual; --50

select round(45.926,-2) from dual; --0

select round(65.926,-2) from dual; --100

--截断

select trunc(45.926,2) from dual; --45.92

select trunc(45.926,1) from dual; -- 45.9

select trunc(45.926,0) from dual; --45

select trunc(45.926,-1) from dual; --40

select trunc(45.926,-2) from dual; --0

select trunc(65.926,-2) from dual; --0

--求余

select mod(9,3) from dual; --0

select mod(9,4) from dual; --1

### 字符函数

-- substr(str1,起始索引,长度)

--注意: 起始索引不管写 0 还是 1 都是从第一个字符开始截取

select substr('abcdefg',0,3) from dual; --abc

select substr('abcdefg',1,3) from dual; --abc

select substr('abcdefg',2,3) from dual; --bcd

--获取字符串长度 24 28

select length('abcdefg') from dual;

--去除字符左右两边的空格

select trim(' hello ') from dual;

--替换字符串

Select replace('hello','l','a') from dual; //heaao

### 日期函数

--查询今天的日期

select sysdate from dual;

--查询3个月后的今天的日期

select add\_months(sysdate,3) from dual;

--查询3天后的日期

select sysdate + 3 from dual;

--查询员工入职的天数

select sysdate - hiredate from emp;

select ceil(sysdate - hiredate) from emp;

--查询员工入职的周数

select (sysdate - hiredate)/7 from emp;

--查询员工入职的月数

select months\_between(sysdate,hiredate) from emp;

--查询员工入职的年份

select months\_between(sysdate,hiredate)/12 from emp;

--转换函数 数值转字符 字符转数值 日期

--字符转数值 to\_number(str) 鸡肋

select 100+'10' from dual; --110 默认已经帮我们转换

select 100 + to\_number('10') from dual; --110

--数值转字符

select to\_char(sal,'$9,999.99') from emp;

select to\_char(sal,'L9,999.99') from emp;

/\*

to\_char(1210.73, '9999.9') 返回 '1210.7'

to\_char(1210.73, '9,999.99') 返回 '1,210.73'

to\_char(1210.73, '$9,999.00') 返回 '$1,210.73'

to\_char(21, '000099') 返回 '000021'

to\_char(852,'xxxx') 返回' 354'

\*/

--日期转字符 to\_char()

select to\_char(sysdate,'yyyy-mm-dd hh:mi:ss') from dual;

select to\_char(sysdate,'yyyy-mm-dd hh24:mi:ss') from dual;

--只想要年

select to\_char(sysdate,'yyyy') from dual; --2017

--只想要日

select to\_char(sysdate,'d') from dual; --2 代表一个星期中第几天

select to\_char(sysdate,'dd') from dual; --10 代表一个月中的第几天

select to\_char(sysdate,'ddd') from dual; --100 代表一年中的第几天

select to\_char(sysdate,'day') from dual; --monday

select to\_char(sysdate,'dy') from dual; --mon 星期的简写

--字符转日期

select to\_date('2017-04-10','yyyy-mm-dd') from dual;

--查询1981年 -- 1985年入职的员工信息

select \* from emp where hiredate between to\_date('1981','yyyy') and to\_date('1985','yyyy');

### 通用函数:

nvl(参数1,参数2) 如果参数1 = null 就返回参数2

nvl2(参数1,参数2,参数3) 如果参数1 = null ,就返回参数3, 否则返回参数2

nullif(参数1,参数2) 如果参数1 = 参数2 那么就返回 null , 否则返回参数1

coalesce: 返回第一个不为null的值

\*/

select nvl2(null,5,6) from dual; --6;

select nvl2(1,5,6) from dual; --5;

select nullif(5,6) from dual; --5

select nullif(6,6) from dual; --null

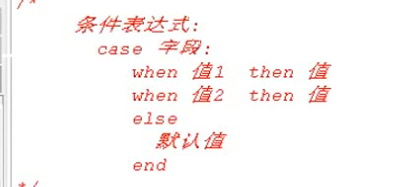
select coalesce(null,null,3,5,6) from dual; --3

select ceil(-12.5) from dual; --12

select floor(12.5) from dual; --12

select ' hello ' from dual;

select \* from emp;



### 条件表达式

|  |
| --- |
| select  case ename  when 值1 then 值  when 值2 then 值  else  默认值  end  from emp; |

--给表中姓名取一个中文名

select

case ename

when ‘SMITH’ then ‘刘备小二’

when ‘ALLEN’ then ‘诸葛村夫’

else

‘路人甲’

end

from emp;

-给表中姓名取一个中文名

select decode(ename,’SMITH’ ’刘备小二’, ’ ALLEN ’ ,’ 诸葛村夫’,’SCOTT’,’曹贼’,’路人乙’) from emp;

### 分组表达式 group by

select 分组后的条件,分组之后的操作 from 表名 group by 分组的条件 having 条件过滤

sql的编写顺序：

select …from…where…group by … having … order by

sql的执行顺序

form…where … group by … having …. select …order by

where和having 的区别

where后面不能接聚合函数，可以接单行函数

having是在group by 之后执行，可以接聚合函数。

---分组统计所有部门的平均工资，找出平均工资大于2000的部门

selectg deptno,avg(sal) from emp group by deptno;

---过滤出大于2000的

select deptno,avg(sal) from emp group by deptno having avg(sal)>2000;

---出错

select deptno,avg(sal) bb from emp group by deptno having bb>2000;

## 第二天: ★★★★★

1.多表查询

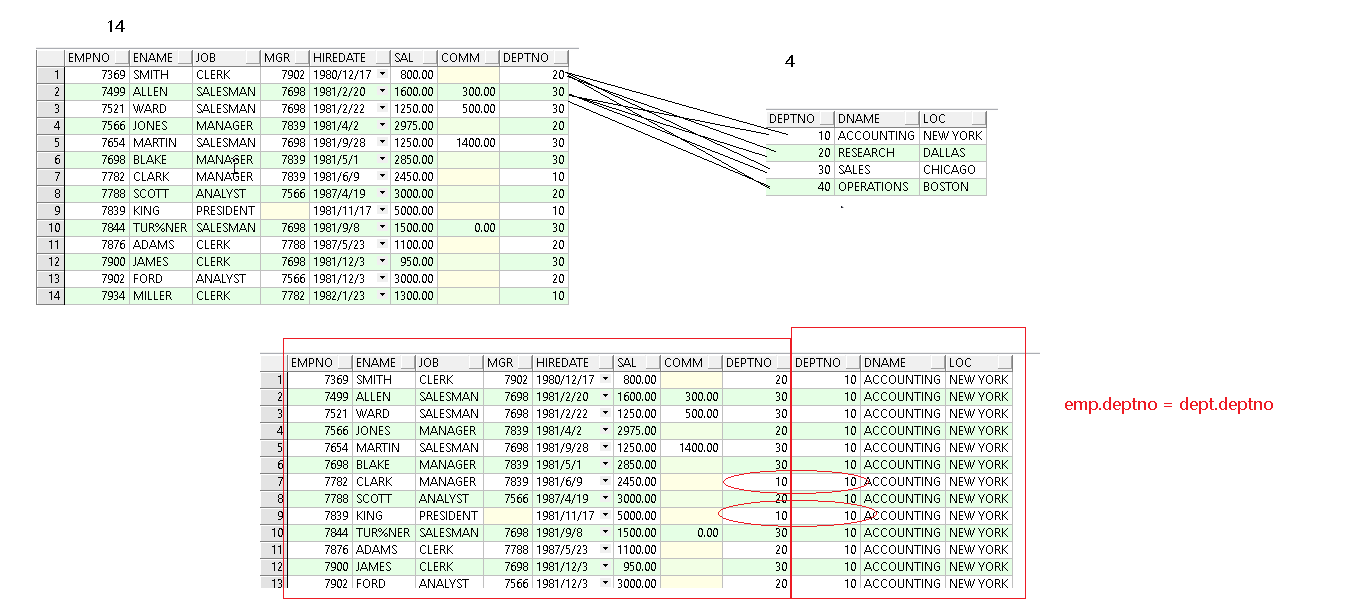
2.子查询

select \* from bonus;

select \* from salgrade;

### 多表查询:

笛卡尔积: 实际上是两张表的乘积,但是在实际开发中没有太大意义



格式: select \* from 表1,表2

\*/

select \* from emp;

select \* from dept;

select \* from emp, dept;

select \* from emp e1, dept d1 where e1.deptno = d1.deptno;

/\*

内联接:

隐式内联接:

等值内联接: where e1.deptno = d1.deptno;

不等值内联接: where e1.deptno <> d1.deptno;

自联接: 自己连接自己

显示内联接:

select \* from 表1 inner join 表2 on 连接条件

inner 关键字可以省略

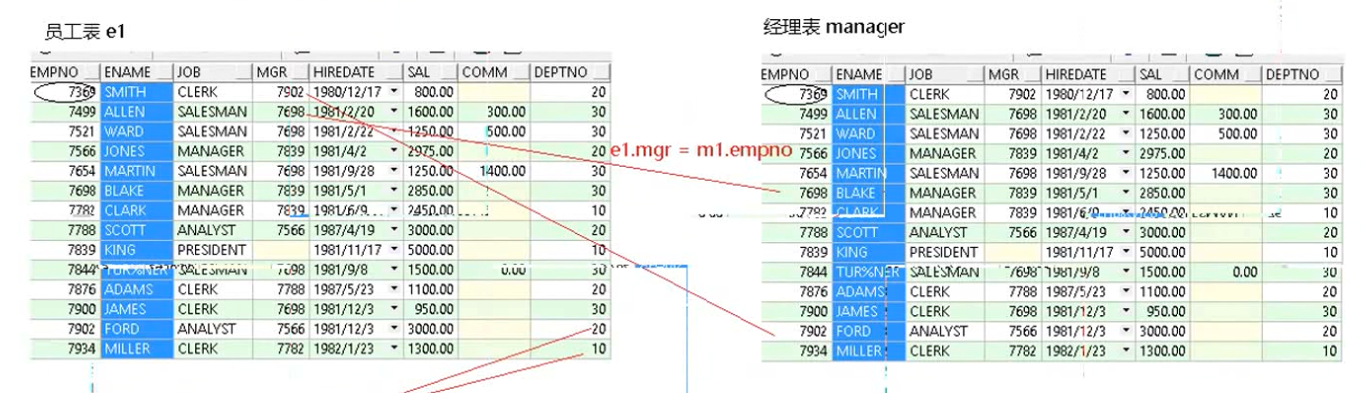
\*/

select \* from emp e1, dept d1 where e1.deptno <> d1.deptno;

--查询员工编号,员工姓名,经理的编号,经理的姓名

select e1.empno,e1.ename,e1.mgr,m1.ename

from emp e1, emp m1 where e1.mgr= m1.empno;



--查询员工编号,员工姓名,员工的部门名称,经理的编号,经理的姓名

select e1.empno,e1.ename,d1.dname,e1.mgr,m1.ename

from emp e1, emp m1,dept d1 where e1.mgr= m1.empno and e1.deptno = d1.deptno;



--查询员工编号,员工姓名,员工的部门名称,经理的编号,经理的姓名,经理的部门名称

select e1.empno,e1.ename,d1.dname,e1.mgr,m1.ename,d2.dname

from emp e1, emp m1,dept d1,dept d2

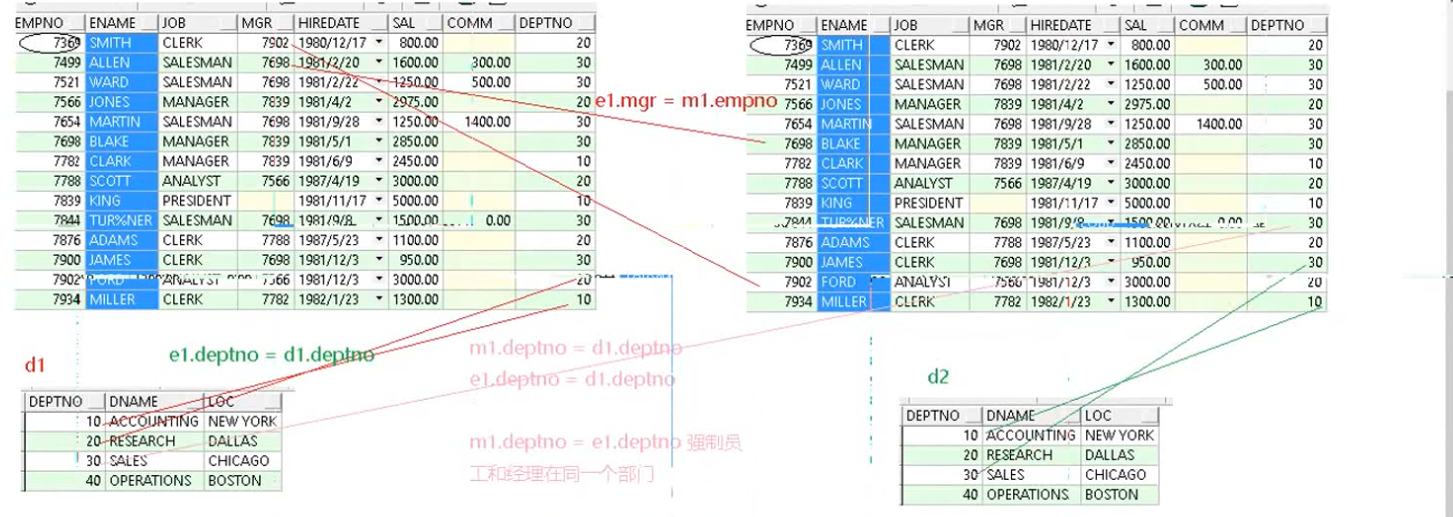
where

e1.mgr= m1.empno

and e1.deptno = d1.deptno

and m1.deptno = d2.deptno

;



--查询员工编号,员工姓名,员工的部门名称,员工的工资等级,经理的编号,经理的姓名,经理的部门名称

select e1.empno,e1.ename,d1.dname,s1.grade,e1.mgr,m1.ename,d2.dname

from emp e1, emp m1,dept d1,dept d2,salgrade s1

where

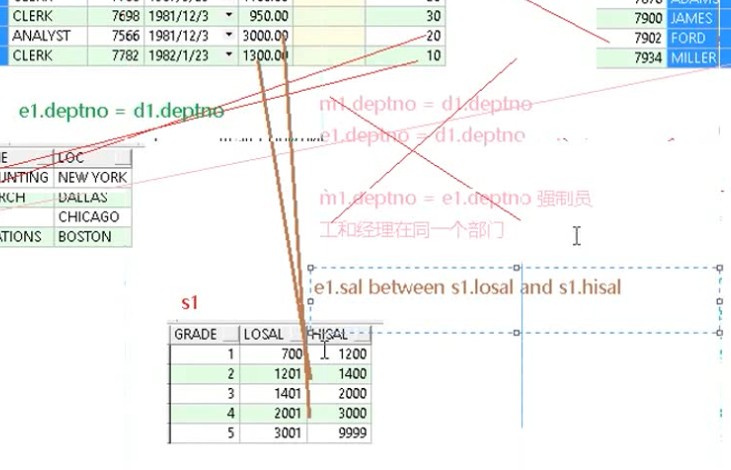
e1.mgr= m1.empno

and e1.deptno = d1.deptno

and m1.deptno = d2.deptno

and e1.sal between s1.losal and s1.hisal

;



--查询员工编号,员工姓名,员工的部门名称,员工的工资等级,经理的编号,经理的姓名,经理的部门名称,经理的工资等级

select e1.empno,e1.ename,d1.dname,s1.grade,e1.mgr,m1.ename,d2.dname,s2.grade

from emp e1, emp m1,dept d1,dept d2,salgrade s1,salgrade s2

where

e1.mgr= m1.empno

and e1.deptno = d1.deptno

and m1.deptno = d2.deptno

and e1.sal between s1.losal and s1.hisal

and m1.sal between s2.losal and s2.hisal

;

--查询员工编号,员工姓名,员工的部门名称,员工的工资等级,经理的编号,经理的姓名,经理的部门名称,经理的工资等级

--将工资等级 1,2,3,4 显示成 中文的 一级 二级 三级...

select e1.empno,

e1.ename,

d1.dname,

case s1.grade

when 1 then '一级'

when 2 then '二级'

when 3 then '三级'

when 4 then '四级'

else

'五级'

end "等级",

e1.mgr,

m1.ename,

d2.dname,

decode(s2.grade,1,'一级',2,'二级',3,'三级',4,'四级','五级') "等级"

from emp e1, emp m1,dept d1,dept d2,salgrade s1,salgrade s2

where

e1.mgr= m1.empno

and e1.deptno = d1.deptno

and m1.deptno = d2.deptno

and e1.sal between s1.losal and s1.hisal

and m1.sal between s2.losal and s2.hisal

;

#### 内连接

--查询员工姓名和员工部门所处的位置

select e1.ename,d1.loc from emp e1,dept d1 where e1.deptno = d1.deptno;

select \* from emp e1 inner join dept d1 on e1.deptno = d1.deptno;

#### 外连接: (标准,通用写法)

左外连接: left outer join 左表中所有的记录,如果右表没有对应记录,就显示空

右外连接: right outer join 右表中的所有记录,如果左表没有对应记录,就显示空

outer 关键字可以省略

Oracle中的外连接: (+) 实际上是如果没有对应的记录就加上空值

select \* from emp e1,dept d1 where e1.deptno = d1.deptno(+);

\*/

select \* from emp e1 left outer join dept d1 on e1.deptno = d1.deptno;

insert into emp(empno,ename) values(9527,'HUAAN');

select \* from emp e1,dept d1 where e1.deptno = d1.deptno(+);

select \* from emp e1 right outer join dept d1 on e1.deptno = d1.deptno;

select \* from emp e1,dept d1 where e1.deptno(+) = d1.deptno;

### 子查询

/\*

子查询: 查询语句中嵌套查询语句; 用来解决复杂的查询语句

查询最高工资的员工信息

单行子查询: > >= = < <= <> !=

多行子查询: in not in >any >all exists not exists

查询领导信息

\*/

--查询最高工资的员工信息

--1.查询出最高工资 --5000

select max(sal) from emp;

--2. 工资等于最高工资

select \* from emp where sal = (select max(sal) from emp);

--查询出比雇员7654的工资高,同时和7788从事相同工作的员工信息

--1.雇员7654的工资 1250

select sal from emp where empno = 7654;

--2.7788从事的工作 ANALYST

select job from emp where empno = 7788;

--3.两个条件合并

select \* from emp where sal > 1250 and job = 'ANALYST';

select \* from emp where sal > (select sal from emp where empno = 7654) and job = (select job from emp where empno = 7788);

--查询每个部门最低工资的员工信息和他所在的部门信息

--1.查询每个部门的最低工资,分组统计

select deptno,min(sal) minsal from emp group by deptno;

--2.员工工资等于他所处部门的最低工资

select \*

from emp e1,

(select deptno,min(sal) minsal from emp group by deptno) t1

where e1.deptno = t1.deptno and e1.sal = t1.minsal;



--3.查询部门相关信息

select \*

from emp e1,

(select deptno,min(sal) minsal from emp group by deptno) t1,

dept d1

where e1.deptno = t1.deptno and e1.sal = t1.minsal and e1.deptno = d1.deptno;

/\*

内联接, 单行子查询, 多行子查询

in

not in

any

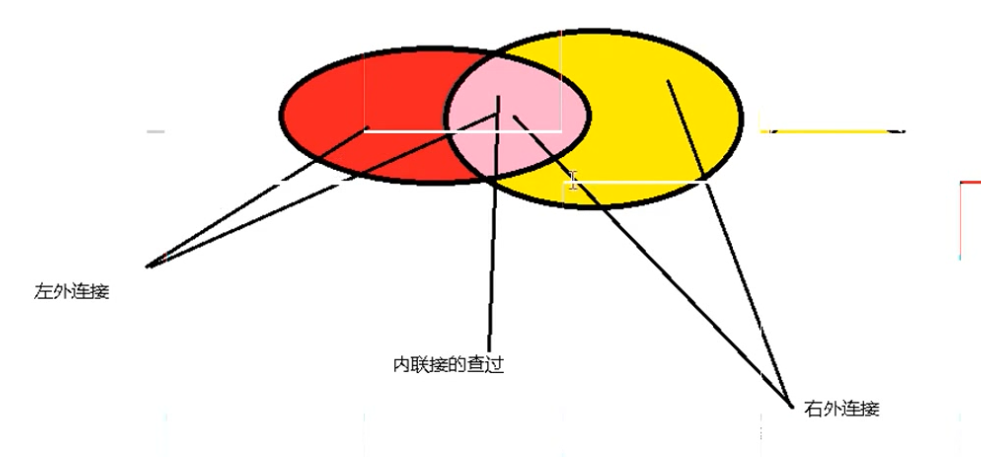
all

exists

通常情况下, 数据库中不要出现null 最好的做法加上Not null

null值并不代表不占空间, char(100) null 100个字符

\*/



--查询领导信息

--1.查询所有经理的编号

select mgr from emp;

select distinct mgr from emp;

--2.结果

select \* from emp where empno in (select mgr from emp);

--查询不是领导的信息

select \* from emp where empno not in (select mgr from emp);

select \* from emp where empno <>all(select mgr from emp);

--正确的写法

select \* from emp where empno not in (select mgr from emp where mgr is not null);

--查询出比10号部门任意一个员工薪资高的员工信息 10 20 30

select \* from emp where sal >any (select sal from emp where deptno = 10);

--查询出比20号部门所有员工薪资高的员工信息 10 20 30

--1.20号最高工资 5000

select max(sal) from emp where deptno =20;

--2.员工信息

select \* from emp where sal > (select max(sal) from emp where deptno =20);

-----使用多行子查询完成上面这题

---------20号部门所有员工薪资 (800 2975 ...)

select sal from emp where deptno = 20;

---------大于集合所有的

select \* from emp where sal >all(select sal from emp where deptno = 20);

### exists

/\*

exists(查询语句) : 存在的意思,判断一张表里面的记录是否存在与另外一张表中

当作布尔值来处理:

当查询语句有结果的时候, 就是返回true

否则返回的是false

数据量比较大的时候是非常高效的

\*/

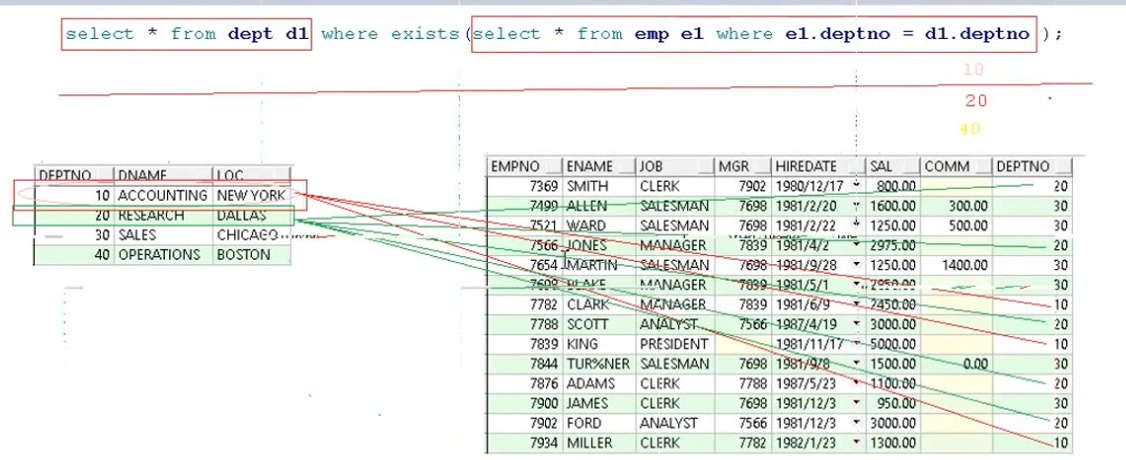
select \* from emp where exists(select \* from emp where deptno = 1234567);

select \* from emp where 3=4;

select \* from emp where exists(select \* from emp where deptno = 20);

--查询有员工的部门的信息

select \* from dept d1 where exists(select \* from emp e1 where e1.deptno = d1.deptno );



--找到员工表中工资最高的前三名(降序排序)

select \* from emp order by sal desc;

/\*

rownum : 伪列, 系统自动生成的一列, 用来表示行号

rownum是Oracle中特有的用来表示行号的, 默认值/起始值是 1 ,在每查询出结果之后,再添加1

rownum最好不能做大于号判断,可以做小于号判断

SQL执行顺序

from .. where ..group by..having .. select..rownum..order by

\*/

Select rownum,e1.\* from emp e1;

--查询rownum大于2的所有记录 ,

select rownum,e1.\* from emp e1 where rownum > 2; --没有任何记录

--查询rownum大于等于1的所有记录

select rownum,e1.\* from emp e1 where rownum >=1;

--查询rownum < 6 的所有记录

select rownum,e1.\* from emp e1 where rownum < 6;

--rownum 排序

select rownum,e1.\* from emp e1 order by sal;

--找到员工表中工资最高的前三名

select e1.\* from emp e1 order by sal desc;

--将上面的结果当作一张表处理,再查询

select rownum, t1.\* from (select e1.\* from emp e1 order by sal desc) t1;

--只要显示前三条记录

select rownum, t1.\* from (select e1.\* from emp e1 order by sal desc) t1 where rownum < 4;

--找到员工表中薪水大于本部门平均薪水的员工

--1.分组统计部门平均薪水

select deptno,avg(sal) avgsal from emp group by deptno;

--2.员工工资 > 本部门平均工资

select \* from emp e1,(select deptno,avg(sal) avgsal from emp group by deptno) t1

where e1.deptno = t1.deptno and e1.sal > t1.avgsal;

/\*

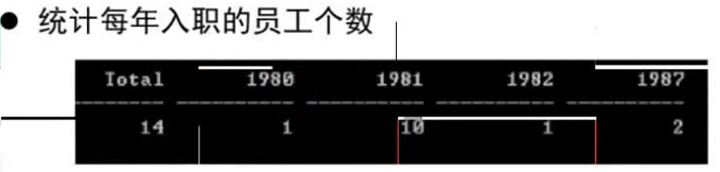
关联子查询 , 非关联子查询

\*/

select \* from emp e where sal > (select avg(sal) from emp e2 group by deptno having e.deptno=e2.deptno);

/\*

统计每年入职的员工个数



\*/

select hiredate from emp;

--只显示年

select to\_char(hiredate,'yyyy') from emp;

--分组统计

select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy');

--将1980竖起来

select yy

from

(select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

select case yy when '1987' then cc end

from

(select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

select case yy when '1987' then cc end "1987"

from

(select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

--去除行记录中的空值

select sum(case yy when '1987' then cc end) "1987"

from

(select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

--统计员工的总数

select sum(cc) "TOTAL"

from

(select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

--将1987 和TOTAL 合并在一起

select

sum(cc) "TOTAL",

sum(case yy when '1987' then cc end) "1987"

from

(select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

--显示所有年份的结果

select

sum(cc) "TOTAL",

sum(case yy when '1980' then cc end) "1980",

sum(case yy when '1981' then cc end) "1981",

sum(case yy when '1982' then cc end) "1982",

sum(case yy when '1987' then cc end) "1987"

from

(select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

/\*

rowid : 伪列 每行记录所存放的真实物理地址

rownum : 行号 , 每查询出记录之后,就会添加一个行号

\*/

select rowid,e.\* from emp e;

--去除表中重复记录

create table p(

name varchar2(10)

);

insert into p values('黄伟福');

insert into p values('赵洪');

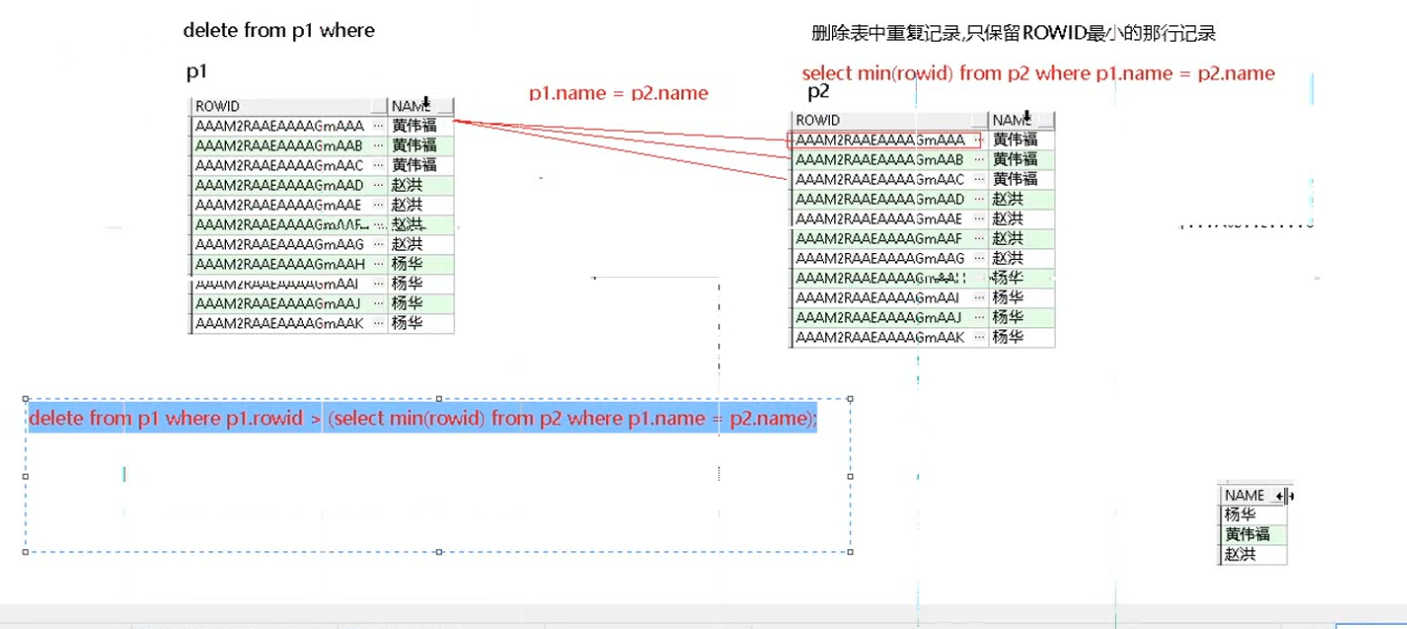
insert into p values('杨华');

delete from p where

select rowid,p.\* from p;

select distinct \* from p;

delete from p p1 where rowid > (select min(rowid) from p p2 where p1.name = p2.name);



/\*

rownum : 分页查询

在oracle中只能使用子查询来做分页查询

\*/

--查询第6 - 第10 记录

select rownum, emp.\* from emp;

select rownum hanghao, emp.\* from emp;

select \* from (select rownum hanghao, emp.\* from emp) tt where tt.hanghao between 6 and 10;

### 集合运算

/\*

集合运算:

并集: 将两个查询结果进行合并

交集

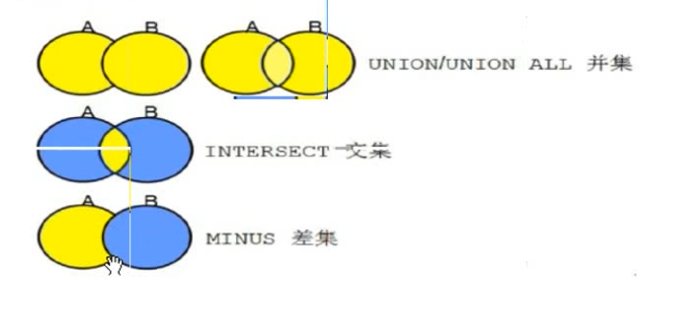
差集

所有的查询结果可能不是来自同一张表,

emp 2000年

2017年 手机 详细信息 emp2017

\*/



--工资大于1500,或者20号部门下的员工

select \* from emp where sal > 1500 or deptno = 20;

--工资大于1500

select \* from emp where sal > 1500;

--20号部门下的员工

select \* from emp where deptno = 20;

--并集运算: union union all

/\*

union : 去除重复的,并且排序

union all : 不会去除重复的

\*/

select \* from emp where sal > 1500

union

select \* from emp where deptno = 20;

select \* from emp where sal > 1500

union all

select \* from emp where deptno = 20;



/\*

交集运算: intersect

\*/

--工资大于1500,并且20号部门下的员工

select \* from emp where sal > 1500;

select \* from emp where deptno = 20;

select \* from emp where sal > 1500

intersect

select \* from emp where deptno = 20;

/\*

差集运算: 两个结果相减

\*/

--1981年入职员工(不包括总裁和经理)

--1981年入职员工

select \* from emp where to\_char(hiredate,'yyyy')='1981';

--总裁和经理

select \* from emp where job = 'PRESIDENT' or job = 'MANAGER';

select \* from emp where to\_char(hiredate,'yyyy')='1981'

minus

select \* from emp where job = 'PRESIDENT' or job = 'MANAGER';

/\*

集合运算中的注意事项:

1.列的类型要一致

2.按照顺序写

3.列的数量要一致,如果不足,用空值填充

\*/

select ename,sal from emp where sal > 1500

union

select ename,sal from emp where deptno = 20;

--列的类型不匹配

select ename,sal from emp where sal > 1500

union

select sal,ename from emp where deptno = 20;

--列的数量不匹配

select ename,sal,deptno from emp where sal > 1500

union

select ename,sal from emp where deptno = 20;

select ename,sal,deptno from emp where sal > 1500

union

select ename,sal,null from emp where deptno = 20;

select ename,sal,deptno from emp where sal > 1500

union

select ename,sal,66 from emp where deptno = 20;

select \* from emp;

select \* from dept;

|  |
| --- |
| --1.列出至少有三个员工的所有部门和部门信息。  select d.\* from dept d,  (select e.deptno, count(1) n from emp e group by e.deptno) t  where d.deptno=t.deptno and t.n>=3;  --2.列出受雇日期早于直接上级的所有员工的编号，姓名，部门名称  select emp\_table.empno, emp\_table.ename, emp\_dept.dname  from emp emp\_table, emp mgr\_table, dept emp\_dept  where emp\_table.mgr=mgr\_table.empno  and emp\_table.deptno=emp\_dept.deptno  and emp\_table.hiredate>mgr\_table.hiredate;  --3.列出职位为“CLERK”的姓名和部门名称，部门人数：  select e.ename, d.dname, t.n  from emp e,  dept d,  (select deptno, count(1) n from emp group by deptno) t  where e.job='CLERK' --确定职位为“CLERK”  and e.deptno=d.deptno  and e.deptno=t.deptno;  --4.列出和“SCOTT”从事相同工作的所有员工及部门名称：  select e.\*, d.dname  from emp e, dept d  where e.job=(select job from emp where ename='SCOTT')  and e.deptno=d.deptno;  --5.列出每个部门工作的员工数量、平均工资和平均服务期限（单位为年）  select e.deptno,  count(1),  avg(e.sal),  to\_char(sysdate, 'yyyy')-round(avg(to\_char(e.hiredate, 'yyyy')))  from emp e  group by e.deptno;  --6、列出各个部门的MANAGER 的最低薪金：  select e.deptno, min(e.sal)  from emp e  where e.job='MANAGER'  group by e.deptno;  --7、给任职日期超过10年的人加薪10%；  update emp t set t.sal=t.sal\*1.1  where trunc(months\_between(sysdate, t.hiredate)/12) > 10;  --17,统计薪资 大于 薪资最高的员工 所在部门 的平均工资 和 薪资最低的员工所在部门的平均工资 的平均工资 的员工信息。  ----1.薪资最高 5000  select max(sal) from emp;  ----2.薪资最高的员工的部门编号 10  select deptno from emp where sal = (select max(sal) from emp);  ----3.部门的平均工资 2916  select avg(sal) from emp where deptno = ( select deptno from emp where sal = (select max(sal) from emp));  ----4.薪资最低 800  select min(sal) from emp;  ----5.薪资最低的员工所在部门 20  select deptno from emp where sal = ( select min(sal) from emp);  ----6.部门的平均工资 2175  select avg(sal) from emp where deptno = (select deptno from emp where sal = ( select min(sal) from emp));  ----7.查询上面两个部门的平均工资 2545.5 2545  select (2916+2175)/2 from dual;    select  (  (select avg(sal) from emp where deptno = ( select deptno from emp where sal = (select max(sal) from emp)))  +  (select avg(sal) from emp where deptno = (select deptno from emp where sal = ( select min(sal) from emp)))  )/2  from dual;  ----8.工资大于 上面两个部门的平均工资  select \* from emp where sal > (select  (  (select avg(sal) from emp where deptno = ( select deptno from emp where sal = (select max(sal) from emp)))  +  (select avg(sal) from emp where deptno = (select deptno from emp where sal = ( select min(sal) from emp)))  )/2  from dual); |

## 第三天:

/\*

多表查询:

内联接

等值联接

inner join

外连接

left outer join

right outer join

Oracle特有的写法(+)

子查询:

\*/

--内联接的结果

select \* from emp e1,emp t1 where e1.mgr = t1.empno;

--t1表里面的所有记录都会显示出来, 如果t1.empno 在e1.mgr中没有对应的记录,就加空值

select \* from emp e1,emp t1 where e1.mgr(+) = t1.empno;

--查询不是领导的员工编号

select empno from emp where empno not in (select mgr from emp where mgr is not null);

--获取员工的名字和部门的名字

select e.ename,d.dname from emp e,dept d where e.deptno = d.deptno;

----使用子查询的方式来完成

select ename,deptno from emp;

select ename,deptno,deptno from emp;

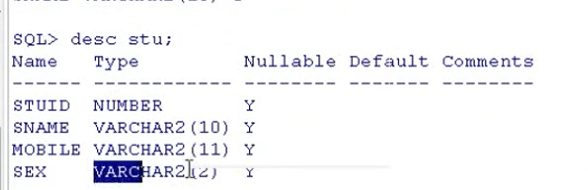
--关联子查询, 子查询依赖外部查询的条件

select e.ename,e.deptno,(select d.dname from dept d where d.deptno = e.deptno ) aa from emp e;



--统计薪资大于薪资最高的员工所在部门的平均工资和薪资最低的员工所在部门的平均工资的平均工资的员工信息。

### 表空间的概念



/\*

Oracle体系结构:

数据库 ---> 数据库实例ORCL ---> 表空间 (用户里面的创建表) ---> 数据文件

地球 ---> 中国 ---> 省份 (人民) ---> 土地山川河流

雄安新区 ---> 人(开发荒地,种地)

创建表空间: 逻辑单位, 通常我们新建一个项目,就会去新建表空间,在表空间中创建用户来创建表

语法:

create tablespace 表空间的名称

datafile '文件的路径(服务器上)'

size 大小

autoextend on 自动扩展

next 每次扩展的大小

\*/

--切换到system帐号下创建

--创建一个表空间 --- 汉东

create tablespace handong

datafile 'c:\handong.dbf'

size 100m

autoextend on

next 10m;

--删除表空间

drop tablespace handong;

/\*

创建用户

create user 用户名

identified by 密码

default tablespace 表空间的名称

\*/

create user dakang

identified by dakang

default tablespace handong;

/\*

授权 grant 角色 | 权限 to 用户

\*/

grant connect to dakang;

--授予 dba的角色

grant dba to dakang;

select \* from scott.emp;

/\*

创建表:

create table 表名(

列名 列的类型 [列的约束],

列名 列的类型 [列的约束]

);

列的类型:

varchar ,在Oracle中,目前是支持的, 但是不保证以后还支持

varchar2(长度) 可变字符长度 varchar2(10) hello 占5个字符

char(长度) 固定长度字符 char(10) hello 占10个字符,用空格填充

number(总长度,小数长度) 数字类型 --小数长度不能大于等于总长度

date 年月日时分秒 2017/4/13 9:43:49

timestamp 时间戳, 比date类型更加精确 13-APR-17 09.44.08.272000 AM +08:00

LONG/CLOB : 存放一本小说

BLOB : 存放电影 java 存进去, 再读取出来

#### 使用子查询的方 式创建表

create table 表名 as 查询语句;

注意: 只会复制表结构和表中的数据,不会复制列的约束

如果查询语句有结果, 就是复制 表结构和数据

如果查询语句没有结果, 就是复制 表结构

\*/

create table test1(

name1 varchar2(10),

name2 char(10),

age number(2,3)

);

insert into test1(name1,name2) values('hello','hello');

select \* from test1 where name1 like 'hello'; --可以查询数据

select \* from test1 where name2 like 'hello'; --查不出数据

insert into test1(age) values(2);

select current\_date from dual;

select current\_timestamp from dual;

select \* from test1;

-- create table 表名 as 查询语句; 复制表

select \* from scott.emp;

create table emp as select \* from scott.emp;

--如果查询语句是没有任何的结果的

select \* from scott.emp where 1=2;

create table emp1 as select \* from scott.emp where 1=2;

select \* from emp1;

/\*

修改表:

添加列

修改列 vharchar2(10)

删除列

修改列名

重命名表

SQL分类:

DDL : 数据定义语言, 修改的结构 alter create drop truncate

DML :　数据操纵语言 , 操作表中数据 insert update delete

DCL : 数据控制语言 , grant

DQL : select

\*/

create table stu(

stuid number,

sname varchar(10)

);

--添加一列

alter table stu add phone varchar2(11);

alter table stu add (

mobile varchar2(11),

sex varchar2(2)

);

--修改列的类型

alter table stu modify sex varchar2(4);

--修改列名 sex --- gender

alter table stu rename column sex to gender;

--删除列

alter table stu drop column gender;

--修改表名

rename stu to student;

--删除表

drop table student;

### 用户

### 序列

### 同义词

/\*

表的五大约束

列的约束: 约束主要是用来约束表中数据的规则

主键约束: primary key 不能为空, 必须唯一

非空约束

唯一约束

检查约束 check(条件) 在mysql中是可以写的,但是mysql直接忽略了检查约束

外键约束:

主要是用来约束从表A中的记录,必须是存在于主表B中

\*/

--男,女,人妖

create table student(

stuid number primary key,

sname varchar2(10) unique,

age varchar2(10) not null,

gender varchar2(4) check( gender in ('男','女','人妖'))

);

--主键约束违反

insert into student values(1,'张三','31','男');

insert into student values(1,'李四','31','男');

--唯一约束违反

insert into student values(1,'徐立','31','男');

insert into student values(2,'徐立','31','男');

--非空约束

insert into student values(1,'徐立',null,'男');

--检查约束

insert into student values(1,'徐立','31','男');

insert into student values(1,'徐立','31','妖');

select \* from student;

/\*

商品分类,商品表

\*/

--商品分类表

create table category(

cid number primary key,

cname varchar2(20)

);

--创建一个商品表

create table product(

pid number primary key,

pname varchar2(20),

cno number

);

insert into category values(1,'手机数码');

insert into product values(10,'锤子',11);

--添加外键约束

alter table product add foreign key(cno) references category(cid);

insert into product values(10,'锤子',11);--插入失败

--1.首先主表中必须存在11号, 先往主表中插入数据,再往从表中插入数据

insert into category values(2,'电脑办公');

insert into product values(11,'外星人',2);

--删除Category

drop table category; --表中记录被外键关联无法删除

--强制删除表(不建议使用) : 先删除外键关联表的外键约束,然后再删除自己, 先删除product的外键约束,再删除category

drop table category cascade constraint;

--级联删除

----添加外键约束,使用级联约束 ,在删除的时候,使用级联删除

alter table product add foreign key(cno) references category(cid) on delete cascade;

insert into category values(2,'电脑办公');

insert into product values(11,'外星人',2);

--级联删除 : 首先去从表中找有没有 关联数据, 如果在从表中找到关联数据,先删除从表中关联数据,然后再删除表中的数据

delete from category where cid = 2;

select \* from category;

select \* from product;

drop table product;

truncate table product;

truncate table category;

/\*

插入数据:

insert into 表名 values(所有列的值都要对应写上)

insert into 表名(列1,列2) values(值1,值2);

使用子查询插入数据

insert into 表名 查询语句

\*/

select \* from emp1;

select \* from emp;

--将emp中10号部门的员工信息,插入到emp1中

insert into emp1 select \* from emp where deptno = 10;

/\*

更新数据

update 表名 set 列名 = 列的值 [where 条件]

\*/

update emp1 set ename='HUAAN' where ename = 'KING';

select \* from emp1;

/\*

删除数据:

delete from 表名 [where 条件]

delete和truncate 区别

delete: truncate:

DML DDL

逐条删除 先删除表再创建表

支持事务操作 不支持事务操作,

执行效率要高

\*/

delete from emp1 where empno=7839;

/\*

事务: 就是一系列的操作,要么都成功,要么都失败

四大特性: 原子性,隔离性,持久性,一致性

如果不考虑隔离级别: 脏读,虚读,不可重复读

MYSQL隔离级别: READ UNCOMMITTED , READ COMMITTED, REPEATABLE READ, SERIALIAZABLE

ORACLE隔离级别: READ COMMITTED SERIALIZABLE READ ONLY

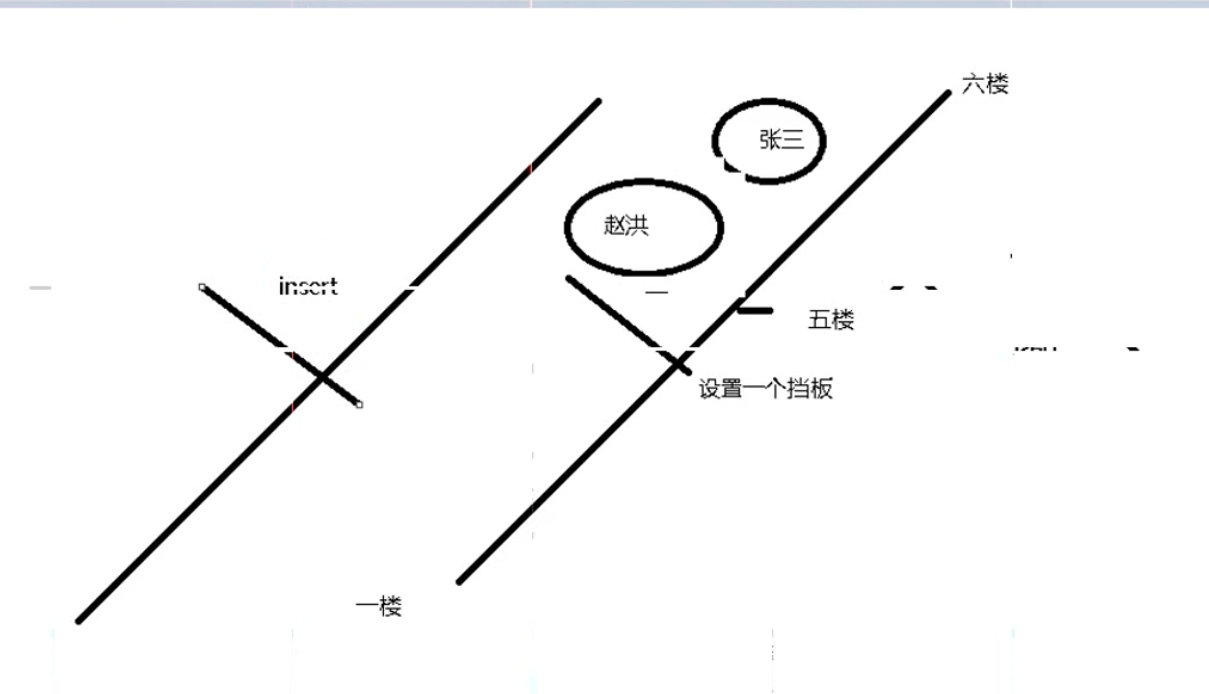
默认隔离级别: READ COMMITTED

提交 : commit

事务的保存点/回滚点: savepoint 保存点的名称

回滚: rollback

\*/



create table louti(

lou number primary key

);

insert into louti values(1);

insert into louti values(2);

insert into louti values(3);

insert into louti values(4);

insert into louti values(5);

savepoint dangban;

insert into louti values(5); --主键约束会发生异常

insert into louti values(6);

rollback to dangban

commit;

declare

begin

insert into louti values(1);

insert into louti values(2);

insert into louti values(3);

insert into louti values(4);

insert into louti values(5);

savepoint dangban;

insert into louti values(5); --这行代码会发生异常

insert into louti values(6);

commit;

exception --捕获异常

when others then

rollback to dangban;

commit;

end;

select \* from louti;

### 视图

/\*

视图: 是对查询结果的一个封装

视图里面所有的数据,都是来自于它查询的那张表,视图本身不存储任何数据

1.能够封装复杂的查询结果

2.屏蔽表中的细节

语法:

create [or replace] view 视图的名称 as 查询语句 [ with read only]

注意: 通常不要通过视图去修改,视图创建的时候,通常要加上with read only

\*/

select \* from emp;

--创建一个视图

create or replace view view\_test1 as select ename,job,mgr from emp;

--通过视图修改数据

update view\_test1 set ename='SMITH2' where ename = 'SMITH';

--创建一个只读视图

create or replace view view\_test2 as select ename,job,mgr from emp with read only;

update view\_test2 set ename='SMITH3' where ename = 'SMITH2';

--视图封装复杂的查询语句

create view view\_test3 as select

sum(cc) "TOTAL",

sum(case yy when '1980' then cc end) "1980",

sum(case yy when '1981' then cc end) "1981",

sum(case yy when '1982' then cc end) "1982",

sum(case yy when '1987' then cc end) "1987"

from

(select to\_char(hiredate,'yyyy') yy,count(1) cc from emp group by to\_char(hiredate,'yyyy')) tt;

--同义词的概念

create synonym dept for view\_test3;

create synonym yuangong for view\_test2;

select \* from yuangong;

select \* from dept;

select \* from view\_test3;

select \* from view\_test2;

/\*

序列: 生成类似于 auto\_increment 这种ID自动增长 1,2,3,4,5....

auto\_increment 这个是mysql

语法:

create sequence 序列的名称

start with 从几开始

increment by 每次增长多少

maxvalue 最大值 | nomaxvalue

minvalue 最小值 | nominvalue

cycle | nocycle 是否循环 1,2,3,1,2,3

cache 缓存的数量3 | nocache 1,2,3,4,5,6

如何从序列获取值

currval : 当前值

nextval : 下一个值

注意: currval 需要在调用nextval之后才能使用

永不回头,往下取数据, 无论发生异常, 回滚

\*/

--创建一个 1,3,5,7,9......30

create sequence seq\_test1

start with 1

increment by 2

maxvalue 30

cycle

cache 10;

select seq\_test1.nextval from dual;

select seq\_test1.currval from dual;

--序列用的最多的一种写法

create sequence seq\_test2;

select seq\_test2.nextval from dual;

create sequence seq\_test3

start with 1

increment by 2

maxvalue 30

minvalue 0

cycle

cache 10;

select seq\_test3.nextval from dual;

### 索引

/\*

索引:相当于是一本书的目录,能够提高我们的查询效率

如果某一列,你经常用来作为查询条件,那么就有必要创建索引,数据量比较的情况

语法:

create index 索引的名称 on 表名(列)

注意:主键约束自带主键索引, 唯一约束自带唯一索引

索引原理: btree balance Tree 平衡二叉树

如果某列作为查询条件的时候,可以提高查询效率,但是修改的时候,会变慢

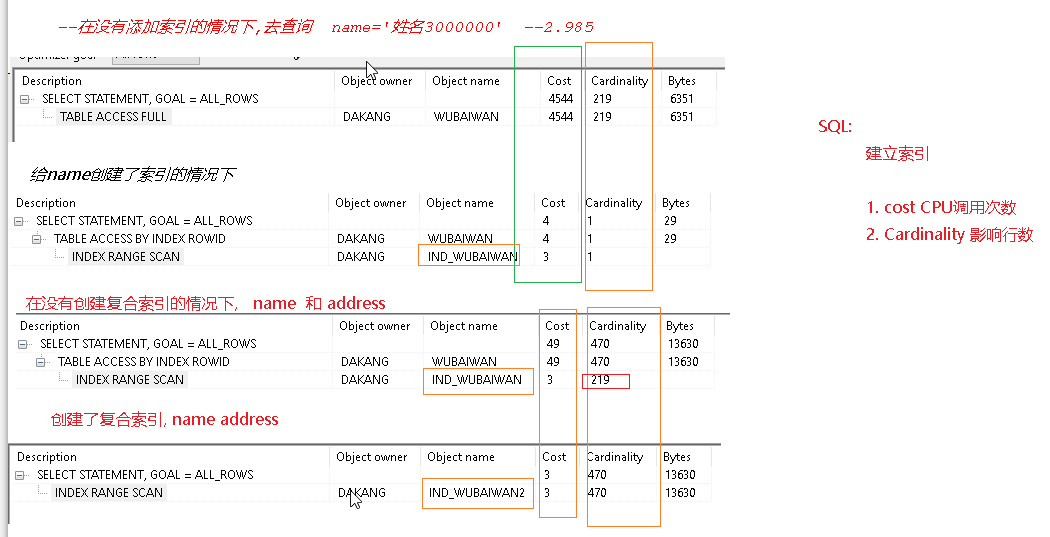
索引创建好之后,过了一段,DBA都会去做重构索引

SQL调优:

1.查看执行计划F5

2. 分析里面的cost 和 影响行数, 想办法降低

\*/



--五百万数据测试

create table wubaiwan(

name varchar2(30),

address varchar2(20)

);

insert into wubaiwan values('')

--插入500000万条数据

declare

begin

for i in 1..5000000 loop

insert into wubaiwan values('姓名'||i,'地址'||i);

end loop;

commit;

end;

--在没有添加索引的情况下,去查询 name='姓名3000000' --2.985

select \* from wubaiwan where name='姓名3000000';

--创建索引 name 再去查询 name='姓名3000000'

create index ind\_wubaiwan on wubaiwan(name);

select \* from wubaiwan where name='姓名3000000'; --0.016

--在没有添加复合索引的情况下,再去查询 name='姓名3000000' and '地址3000000'

select \* from wubaiwan where name='姓名3000000' and address='地址3000000'; --0.032

--创建复合索引的情况下, 再去查询

create index ind\_wubaiwan2 on wubaiwan(name,address);

select \* from wubaiwan where name='姓名3000000' and address='地址3000000'; --0.015

/\*

DDL表空间操作

创建表空间

创建用户

授权

创建表

子查询创建表

修改表 : 添加列,删除列,修改列,修改列名, 修改表名

约束:

主键约束,唯一约束,非空约束,检查约束,外键约束

外键约束:

强制删除

级联删除

DML表中数据:

插入数据

子查询插入数据

更新数据

删除数据: delete 和 truncate

事务操作:

savepoint 保存点

rollback to 保存点

ORACLE事务隔离级别 : READ COMMITTED

视图: 就像窗户一样, 封装查询结果 , 通常视图创建只读视图

序列: 主要是用来实现ID自增长

索引: 相当于是书的目录,能够提高查询效率, 原理 平衡二叉树, 每隔一段时间DBA都需要去重建索引

同义词: create synonym 名称 for 对象的名称

\*/

|  |
| --- |
| --1.列出至少有三个员工的所有部门和部门信息。  --分组统计每个部门的员工数量  select deptno,count(1) from emp group by deptno;  --找出员工数量至少3    --2.列出受雇日期早于直接上级的所有员工的编号，姓名，部门名称  select emp\_table.empno, emp\_table.ename, emp\_dept.dname  from emp emp\_table, emp mgr\_table, dept emp\_dept  where emp\_table.mgr=mgr\_table.empno  and emp\_table.deptno=emp\_dept.deptno  and emp\_table.hiredate>mgr\_table.hiredate;  --3.列出职位为“CLERK”的姓名和部门名称，部门人数：  select e.ename, d.dname, t.n  from emp e,  dept d,  (select deptno, count(1) n from emp group by deptno) t  where e.job='CLERK' --确定职位为“CLERK”  and e.deptno=d.deptno  and e.deptno=t.deptno;  --4.列出和“SCOTT”从事相同工作的所有员工及部门名称：  select e.\*, d.dname  from emp e, dept d  where e.job=(select job from emp where ename='SCOTT')  and e.deptno=d.deptno;  --5.列出每个部门工作的员工数量、平均工资和平均服务期限（单位为年）  select e.deptno,  count(1),  avg(e.sal),  to\_char(sysdate, 'yyyy')-round(avg(to\_char(e.hiredate, 'yyyy')))  from emp e  group by e.deptno;  --6、列出各个部门的MANAGER 的最低薪金：  select e.deptno, min(e.sal)  from emp e  where e.job='MANAGER'  group by e.deptno;  --7、给任职日期超过10年的人加薪10%；  update emp t set t.sal=t.sal\*1.1  where trunc(months\_between(sysdate, t.hiredate)/12) > 10;  --17,统计薪资 大于 薪资最高的员工 所在部门 的平均工资 和 薪资最低的员工所在部门的平均工资 的平均工资 的员工信息。  ----1.薪资最高 5000  select max(sal) from emp;  ----2.薪资最高的员工的部门编号 10  select deptno from emp where sal = (select max(sal) from emp);  ----3.部门的平均工资 2916  select avg(sal) from emp where deptno = ( select deptno from emp where sal = (select max(sal) from emp));  ----4.薪资最低 800  select min(sal) from emp;  ----5.薪资最低的员工所在部门 20  select deptno from emp where sal = ( select min(sal) from emp);  ----6.部门的平均工资 2175  select avg(sal) from emp where deptno = (select deptno from emp where sal = ( select min(sal) from emp));  ----7.查询上面两个部门的平均工资 2545.5 2545  select (2916+2175)/2 from dual;    select  (  (select avg(sal) from emp where deptno = ( select deptno from emp where sal = (select max(sal) from emp)))  +  (select avg(sal) from emp where deptno = (select deptno from emp where sal = ( select min(sal) from emp)))  )/2  from dual;  ----8.工资大于 上面两个部门的平均工资  select \* from emp where sal > (select  (  (select avg(sal) from emp where deptno = ( select deptno from emp where sal = (select max(sal) from emp)))  +  (select avg(sal) from emp where deptno = (select deptno from emp where sal = ( select min(sal) from emp)))  )/2  from dual); |

### PLSQL编程

/\*

PLSQL编程 : procedure Language 过程语言 Oracle对SQL的一个扩展

让我们能够像在java中一样写 if else else if 条件, 还可以编写循环逻辑 for while

declare

--声明变量

变量名 变量类型;

变量名 变量类型 := 初始值;

vsal emp.sal%type; --引用型的变量

vrow emp%rowtype; --声明记录型变量

begin

--业务逻辑

end;

dbms\_output.put\_line()相当于java中 syso

\*/

declare

i varchar2(10) := '张三';

begin

dbms\_output.put\_line(i);

end;

--查询7369的工资,并且打印出来

declare

vsal emp.sal%type;

begin

--将查询出的结果赋值给vsal

select sal into vsal from emp where empno = 7369;

dbms\_output.put\_line(vsal);

end;

--查询7369的员工信息,并且打印出来

select \* from emp where empno = 7369;

declare

vrow emp%rowtype;

begin

select \* into vrow from emp where empno = 7369;

dbms\_output.put\_line('姓名:'||vrow.ename || '工资'|| vrow.sal);

end;

/\*

PL条件判断

if then

elsif then

else

end if;

\*/

--根据不同年纪,输出相关内容

declare

age number := &aaa;

begin

if age < 18 then

dbms\_output.put\_line('小屁孩');

elsif age>=18 and age <=24 then

dbms\_output.put\_line('年轻人');

elsif age>24 and age < 40 then

dbms\_output.put\_line('老司机');

else

dbms\_output.put\_line('老年人');

end if;

end;

/\*

循环操作

while 循环

while 条件 loop

end loop;

for循环

for 变量 in [reverse] 起始值..结束值 loop

end loop;

loop循环

loop

exit when 条件

end loop;

\*/

--输出1~10

declare

i number :=1;

begin

while i<=10 loop

dbms\_output.put\_line(i);

i := i+1;

end loop;

end;

--输出1~10

declare

begin

for i in reverse 1..10 loop

dbms\_output.put\_line(i);

end loop;

end;

--输出1~10

declare

i number :=1;

begin

loop

exit when i>10;

dbms\_output.put\_line(i);

i := i+1;

end loop;

end;

/\*

\*

\*\*\*

\*\*\*\*\*

\*\*\*

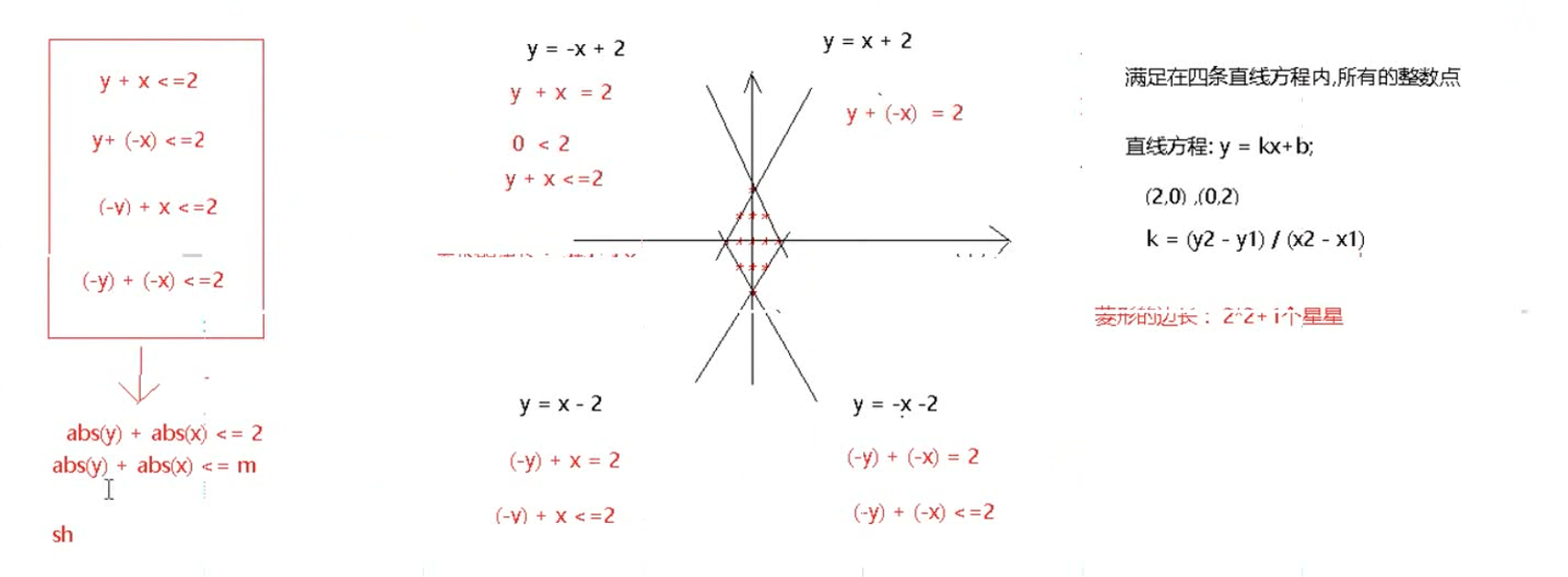
\*

输出 m

x : [-m,m]

y : [-m,m]

输出所有满足条件的 : abs(y)+abs(x) <=m



m取值

\*/

--使用PLSQL输出菱形

declare

m number := 10;

begin

for x in -m..m loop

for y in -m..m loop

if abs(y) + abs(x) <= m then

dbms\_output.put('\*');

else

dbms\_output.put(' ');

end if;

end loop;

dbms\_output.new\_line();

end loop;

end;

--使用PLSQL输出三角形,只要是三个角

declare

m number := 10;

begin

for x in reverse -m..m loop

for y in -m..m loop

if abs(y) + abs(x) <= m and x>=0 then

dbms\_output.put('\*');

else

dbms\_output.put(' ');

end if;

end loop;

dbms\_output.new\_line();

end loop;

end;

## 第四天:

### 1.PLSQL编程

select \* from emp;

-- 序列: ORACLE使用来模拟ID自动增长的

create sequence seq\_test2;

drop table test2;

create table test2(

tid number primary key,

tname varchar2(10)

)

insert into test2 values(seq\_test2.nextval,'张三');

select \* from test2;

/\*

PLSQL编程: 过程语言,编写一些复杂业务逻辑

输出星号:

abs(y) + abs(x) <= m

vsal emp.sal%type --引用型变量

row emp%rowtype --记录型变量

select sal into vsal from emp where empno=7788;

\*/

declare

m number :=5;

begin

for y in -m..m loop

for x in -m..m loop

if abs(x)+abs(y)<=m then

dbms\_output.put('\*');

else

dbms\_output.put(' ');

end if;

end loop;

dbms\_output.new\_line();

end loop;

end;

------------------------------游标:用来操作查询的结果集.相当于JDBC中的ResultSet-------------

declare

current vrows is select \* from emp;--创建游标/储存结果集

--声明变量,操作结果集

vrow emp%rowtype;

begin

open vrows;--打开游标

loop

fetch vrows into vrow;

exit when vrows%notfound;

dbms\_output.put\_line('姓名:'||vrow.ename||'工资'||vrow.sal);

end loop;

close vrows;--关闭游标

end;

--使用游标查询指定10号部门

declare

cursor vrow(dno number) is select \* from emp where deptno = dno;

vrows emp%rowtype;

begin

open vrow(10);

loop

fetch vrow into vrows;

exit when vrow%notfound;

dbms\_output.put\_line('姓名'||vrows.ename||'工资'||vrows.sal);

end loop;

close vrow;

end;

--使用for遍历游标

declare

cursor vrows is select \* from emp;

begin

for vrow in vrows loop

dbms\_output.put\_line('姓名'||vrow.ename||'工资'||vrow.sal);

end loop;

end;

/\*

序列: ORACLE使用来模拟ID自动增长的

\*/

create sequence seq\_test4;

create table test2(

tid number primary key,

tname varchar2(10)

);

insert into test2 values(seq\_test4.nextval,'张三');

select \* from test2;

/\*

PLSQL编程: 过程语言,编写一些复杂业务逻辑

输出星号:

abs(y) + abs(x) <= m

vsal emp.sal%type --引用型变量

row emp%rowtype --记录型变量

select sal into vsal from emp where empno=7788;

\*/

declare

m number := 3;

begin

for y in -m..m loop

for x in -m..m loop

if abs(y) + abs(x) <= m then

dbms\_output.put('\*');

else

dbms\_output.put(' ');

end if;

end loop;

dbms\_output.new\_line();

end loop;

end;

### 2.游标

/\*

游标(光标): 是用来操作查询结果集,相当于是JDBC中ResultSet

语法: cursor 游标名[(参数名 参数类型)] is 查询结果集

开发步骤:

1. 声明游标

2. 打开游标 open 游标名

3. 从游标中取数据 fetch 游标名 into 变量

游标名%found :找到数据

游标名%notfound : 没有找到数据

4. 关闭游标 close 游标名

系统引用游标

1. 声明游标 : 游标名 sys\_refcursor

2. 打开游标: open 游标名 for 结果集

3. 从游标中取数据

4. 关闭游标

for循环遍历游标:

不需要声明额外变量

不需要打开游标

不需要关闭游标

\*/

--输出员工表中所有的员工姓名和工资(不带参数游标)

/\*

游标:所有员工

声明一个变量,用来记录一行数据 %rowtype

\*/

declare

--游标

cursor vrows is select \* from emp;

--s声明变量,记录一行数据

vrow emp%rowtype;

begin

--1.打开游标

open vrows;

--2.从游标提取数据

--循环取数据

loop

fetch vrows into vrow;

exit when vrows%notfound;

dbms\_output.put\_line('姓名:'||vrow.ename ||' 工资: ' || vrow.sal);

end loop;

--3.关闭游标

close vrows;

end;

--输出指定部门下的员工姓名和工资

/\*

游标: 指定部门的所有员工

声明一个变量记录一行数据

\*/

declare

--声明游标

cursor vrows(dno number) is select \* from emp where deptno = dno;

--声明变量

vrow emp%rowtype;

begin

--1.打开游标 , 指定10号部门

open vrows(10);

--2. 循环遍历,取数据

loop

fetch vrows into vrow;

exit when vrows%notfound;

dbms\_output.put\_line('姓名:'||vrow.ename ||' 工资: ' || vrow.sal);

end loop;

close vrows;

end;

--系统引用游标

--输出员工表中所有的员工姓名和工资

declare

--声明系统引用游标

vrows sys\_refcursor;

--声明一个变量

vrow emp%rowtype;

begin

--1.打开游标

open vrows for select \* from emp;

--2.取数据

loop

fetch vrows into vrow;

exit when vrows%notfound;

dbms\_output.put\_line('姓名:'||vrow.ename ||' 工资: ' || vrow.sal);

end loop;

close vrows;

end;

--扩展内容----使用for循环遍历游标

declare

--声明一个游标

cursor vrows is select \* from emp;

begin

for vrow in vrows loop

dbms\_output.put\_line('姓名:'||vrow.ename ||' 工资: ' || vrow.sal || '工作:'|| vrow.job);

end loop;

end;

select \* from emp;

--按照员工工作给所有员工涨工资,总裁涨1000,经理涨800,其他人涨400

/\*

游标 : 所有员工

声明一个记录一行数据

\*/

declare

--声明游标

cursor vrows is select \* from emp;

--声明一个变量

vrow emp%rowtype;

begin

--1.打开游标

open vrows;

--2.循环取数据

loop

--取数据

fetch vrows into vrow;

--退出条件

exit when vrows%notfound;

--根据不同的职位,涨工资 总裁涨1000,经理涨800,其他人涨400

if vrow.job = 'PRESIDENT' then

update emp set sal = sal + 1000 where empno = vrow.empno;

elsif vrow.job = 'MANAGER' then

update emp set sal = sal + 800 where empno = vrow.empno;

else

update emp set sal = sal + 400 where empno = vrow.empno;

end if;

end loop;

--3.关闭游标

close vrows;

--4.提交事务

commit;

end;

select \* from emp;

/\*

例外:(意外)程序运行的过程发生异常,相当于是JAVA中的异常

declare

--声明变量

begin

--业务逻辑

exception

--处理异常

when 异常1 then

...

when 异常2 then

...

when others then

...处理其它异常

end;

zero\_divide : 除零异常

value\_error : 类型转换异常

too\_many\_rows : 查询出多行记录,但是赋值给了rowtype记录一行数据变量

no\_data\_found : 没有找到数据

自定义异常:

异常名 exception;

raise 异常名

\*/

declare

vi number;

vrow emp%rowtype;

begin

--vi := 8/0;

--vi := 'aaa';

--select \* into vrow from emp;

select \* into vrow from emp where empno=1234567;

exception

when zero\_divide then

dbms\_output.put\_line('发生了除零异常');

when value\_error then

dbms\_output.put\_line('发生了类型转换异常');

when too\_many\_rows then

dbms\_output.put\_line(' 查询出多行记录,但是赋值给了rowtype记录一行数据变量');

when no\_data\_found then

dbms\_output.put\_line('没有找到数据异常');

when others then

dbms\_output.put\_line('发生了其它异常' || sqlerrm);

end;

--查询指定编号的员工,如果没有找到,则抛出自定义的异常

/\*

--错误的演示

1.声明一个变量 %rowtype

2.查询员工信息,保存起来

3.判断员工信息是否为空

4. 如果是 则抛出异常

\*/

declare

-- 1.声明一个变量 %rowtype

vrow emp%rowtype;

--2 .声明一个自定义的异常

no\_emp exception;

begin

--查询员工信息,保存起来

select \* into vrow from emp where empno = 8888; --抛出异常

if vrow.sal is null then

raise no\_emp; --抛出自定义的异常

end if;

exception

when no\_emp then

dbms\_output.put\_line('输出了自定义的异常');

when others then

dbms\_output.put\_line('输出了其它异常'||sqlerrm);

end;

--查询指定编号的员工,如果没有找到,则抛出自定义的异常

/\*

游标来判断

%found %notfound

声明一个游标

声明一个变量,记录数据

从游标中取记录

如果有,则不管它

如果没有就抛出自定义的异常

\*/

declare

--声明游标

cursor vrows is select \* from emp where empno=8888;

--声明一个记录型变量

vrow emp%rowtype;

--声明一个自定义异常

no\_emp exception;

begin

--1.打开游标

open vrows;

--2.取数据

fetch vrows into vrow;

--3.判断游标是否有数据

if vrows%notfound then

raise no\_emp;

end if;

close vrows;

exception

when no\_emp then

dbms\_output.put\_line('发生了自定义的异常');

end;

### 3.存储过程

/\*

存储过程: 实际上是封装在服务器上一段PLSQL代码片断,已经编译好了的代码

1.客户端取调用存储过程,执行效率就会非常高效

语法:

create [or replace] procedure 存储过程的名称(参数名 in|out 参数类型,参数名 in|out 参数类型)

is | as

--声明部分

begin

--业务逻辑

end;

\*/

--给指定员工涨薪,并打印涨薪前和涨薪后的工资

/\*

参数 : in 员工编号

参数 : in 涨多少

声明一个变量 : 存储涨工资前的工资

查询出当前是多少

打印涨薪前的工资

更新工资

打印涨薪后的工资

\*/

create or replace procedure proc\_updatesal(vempno in number,vnum in number)

is

--声明变量.记录当前工资

vsal number;

begin

--查询当前的工资

select sal into vsal from emp where empno = vempno;

--输出涨薪前的工资

dbms\_output.put\_line('涨薪前:'||vsal);

--更新工资

update emp set sal = vsal + vnum where empno = vempno;

--输出涨薪后的工资

dbms\_output.put\_line('涨薪后:'||(vsal+vnum));

--提交

commit;

end;

--方式1

call proc\_updatesal(7788,10);

--方式2 用的最多的方式

declare

begin

proc\_updatesal(7788,-100);

end;

### 4.存储函数

/\*

存储函数: 实际上是一段封装是Oracle服务器中的一段PLSQL代码片断,它是已经编译好了的代码片段

语法:

create [or replace] function 存储函数的名称(参数名 in|out 参数类型,参数名 in|out 参数类型) return 参数类型

is | as

begin

end;

存储过程和函数的区别:

1.它们本质上没有区别

2.函数存在的意义是给过程调用 存储过程里面调用存储函数

3.函数可以在sql语句里面直接调用

4.存储过程能实现的,存储函数也能实现,存储函数能实现的,过程也能实现

默认是 in

\*/

--查询指定员工的年薪

/\*

参数 : 员工的编号

返回 : 年薪

\*/

create or replace function func\_getsal(vempno number) return number

is

--声明变量.保存年薪

vtotalsal number;

begin

select sal\*12 + nvl(comm,0) into vtotalsal from emp where empno = vempno;

return vtotalsal;

end;

--调用存储函数

declare

vsal number;

begin

vsal := func\_getsal(7788);

dbms\_output.put\_line(vsal);

end;

--查询员工的姓名,和他的年薪

select ename,func\_getsal(empno) from emp;

--查询员工的姓名和部门的名称

--查询指定员工的年薪--存储过程来实现

--参数: 员工编号

--输出: 年薪

create or replace procedure proc\_gettotalsal(vempno in number,vtotalsal out number)

is

begin

select sal\*12 + nvl(comm,0) into vtotalsal from emp where empno = vempno;

end;

declare

vtotal number;

begin

proc\_gettotalsal(7788,vtotal);

dbms\_output.put\_line('年薪:'||vtotal);

end;

select \* from emp where empno = 8888;

/\*

JAVA调用存储过程

JDBC的开发步骤:

1.导入驱动包

2.注册驱动

3.获取连接

4.获取执行SQL的statement

5.封装参数

6.执行SQL

7.获取结果

8.释放资源

\*/

/\*

封装一个存储过程 : 输出所有表中的记录

输出类型 : 游标

\*/

create or replace procedure proc\_getemps(vrows out sys\_refcursor)

is

begin

--1.打开游标, 给游标赋值

open vrows for select \* from emp;

end;

|  |
| --- |
| package com.itheima.test;  import java.sql.CallableStatement;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.ResultSet;  import org.junit.Test;  import oracle.jdbc.OracleCallableStatement;  import oracle.jdbc.driver.OracleTypes;  /\*  1.导入驱动包  2.注册驱动  3.获取连接  4.获取执行SQL的statement  5.封装参数  6.执行SQL  7.获取结果  8.释放资源  \*/  public class TestProcedure {  @Test  /\*  create or replace procedure proc\_gettotalsal(vempno in number,vtotalsal out number)  is    begin  select sal\*12 + nvl(comm,0) into vtotalsal from emp where empno = vempno;  end;  \* \*/  public void test1() throws Exception{  //注册驱动  Class.forName("oracle.jdbc.driver.OracleDriver");  //2.获取连接  String url = "jdbc:oracle:thin:@192.168.80.100:1521:orcl";  String username = "dakang";  String password = "dakang";  Connection conn = DriverManager.getConnection(url, username, password);  //3.获取执行SQL的statement.这是一个固定格式 proc\_gettotalsal是一个存储函数  String sql = "{call proc\_gettotalsal(?,?)}";  CallableStatement state = conn.prepareCall(sql);  //设置输入参数  state.setInt(1, 7788);//设置员工编号  //注册输出参数类型  state.registerOutParameter(2, OracleTypes.NUMBER);    //4.执行statement  state.execute();    //5.获取执行结果  int totalsal = state.getInt(2);    //输出结果  System.out.println("工资:"+ totalsal);    //6.释放资源  state.close();  conn.close();  }    //调用存储函数  /\*  create or replace function func\_getsal(vempno number) return number  is  --声明变量.保存年薪  vtotalsal number;  begin  select sal\*12 + nvl(comm,0) into vtotalsal from emp where empno = vempno;  return vtotalsal;  end;  \*/  @Test  public void test2() throws Exception{  //注册驱动  Class.forName("oracle.jdbc.driver.OracleDriver");  //2.获取连接  String url = "jdbc:oracle:thin:@192.168.80.100:1521:orcl";  String username = "dakang";  String password = "dakang";  Connection conn = DriverManager.getConnection(url, username,password);  //3.获取执行SQL的statement  String sql = " {?= call func\_getsal(?)}";  CallableStatement state = conn.prepareCall(sql);  //4.封装参数  //注册返回类型参数  state.registerOutParameter(1, OracleTypes.NUMBER);  //设置第二个参数  state.setInt(2, 7788);  //5.执行SQL  state.execute();  //6.获取结果  int totalsal = state.getInt(1);  System.out.println("年薪 : ====" +totalsal);  //7.释放资源  state.close();  conn.close();  }    /\*  create or replace procedure proc\_getemps(vrows out sys\_refcursor)  is    begin  --1.打开游标, 给游标赋值  open vrows for select \* from emp;  end;  \* \*/  @Test  public void test3() throws Exception{  //注册驱动  Class.forName("oracle.jdbc.driver.OracleDriver");  //2.获取连接  String url = "jdbc:oracle:thin:@192.168.80.100:1521:orcl";  String username = "dakang";  String password = "dakang";  Connection conn = DriverManager.getConnection(url, username,password);  //3.获取执行SQL的statement  String sql = "{call proc\_getemps(?)}";  CallableStatement call = conn.prepareCall(sql);  //接口 --- > 对象 -->实现类的名称  System.out.println(call.getClass().getName());  OracleCallableStatement oracleCall = (OracleCallableStatement)call;  //4.注册输出类型的参数  call.registerOutParameter(1, OracleTypes.CURSOR);  //5.执行SQL  call.execute();  //6.获取执行的结果  ResultSet resultSet = oracleCall.getCursor(1);  while(resultSet.next()){  int empno = resultSet.getInt("empno");  String name = resultSet.getString("ename");  System.out.println(empno +" ==== "+name);  }    //7.释放资源  resultSet.close();  call.close();  conn.close();  }  } |

/\*

触发器: 当用户执行了 insert | update | delete 这些操作之后, 可以触发一系列其它的动作/业务逻辑

作用 :

在动作执行之前或者之后,触发业务处理逻辑

插入数据,做一些校验

语法:

create [or replace] trigger 触发器的名称

before | after

insert | update | delete

on 表名

[for each row]

declare

begin

end;

触发器的分类:

语句级触发器: 不管影响多少行, 都只会执行一次

行级触发器: 影响多少行,就触发多少次

:old 代表旧的记录, 更新前的记录

:new 代表的是新的记录

\*/

--新员工入职之后,输出一句话: 欢迎加入黑马程序员

create or replace trigger tri\_test1

after

insert

on emp

declare

begin

dbms\_output.put\_line('欢迎加入黑马程序员');

end;

insert into emp(empno,ename) values(9527,'HUAAN');

--数据校验, 星期六老板不在, 不能办理新员工入职

--在插入数据之前

--判断当前日期是否是周六

--如果是周六,就不能插入

create or replace trigger tri\_test2

before

insert

on emp

declare

--声明变量

vday varchar2(10);

begin

--查询当前

select trim(to\_char(sysdate,'day')) into vday from dual;

--判断当前日期:

if vday = 'saturday' then

dbms\_output.put\_line('老板不在,不能办理入职');

--抛出系统异常

raise\_application\_error(-20001,'老板不在,不能办理入职');

end if;

end;

insert into emp(empno,ename) values(9528,'HUAAN2');

--更新所有的工资 输出一句话

create or replace trigger tri\_test3

after

update

on emp

for each row

declare

begin

dbms\_output.put\_line('更新了数据');

end;

update emp set sal = sal+10;

--判断员工涨工资后的工资一定要大于涨工资前的工资

/\*

200 --> 100

触发器 : before

旧的工资

新的工资

如果旧的工资大于新的工资 , 抛出异常,不让它执行成功

触发器中不能提交事务,也不能回滚事务

\*/

create or replace trigger tri\_updatesal

before

update

on emp

for each row

declare

begin

if :old.sal > :new.sal then

raise\_application\_error(-20002,'旧的工资不能大于新的工资');

end if;

end;

update emp set sal = sal + 10;

select \* from emp;

update emp set sal = sal - 100;

/\*

模拟mysql中ID的自增属性 auto\_increment

insert into person(null,'张三');

触发器:

pid=1 insert pid=1

序列 : create sequence seq\_person\_pid;

\*/

create table person(

pid number primary key,

pname varchar2(20)

);

insert into person values(null,'张三');

create sequence seq\_person\_pid;

--触发器

create or replace trigger tri\_add\_person\_pid

before

insert

on person

for each row

declare

begin

dbms\_output.put\_line(:new.pname);

--给新记录 pid 赋值

select seq\_person\_pid.nextval into :new.pid from dual;

end;

insert into person values(null,'张三');

select \* from person;

### 5.触发器